Migration20

After installing 2.0, the following should be done to migrate old code to new format:
Warning!!! It is not possible to revert migrated models back to old format!

Automatic migration

Just execute Main Menu -> Tools -> Start Migration to MPS 2.0 and follow the instructions.

Non-automatic migration

If you want full control over what MPS does, you can use this way of performing migration. In the next part we'll consider all the refactorings separately.

Do the following:

1. Main Menu -> Tools -> Migrations 2.0 -> Upgrade Persistence
2. Main Menu -> Tools -> Migrations 2.0 -> Fix Module Dependencies
3. Main Menu -> Tools -> Migrations 2.0 -> Migrate Stub Models
4. Open logical view, right-click project, select "Optimize Imports"
5. Main Menu -> Tools -> Scripts -> All Scripts..., check all scripts with type "Migration (MPS 2.0)", click "Run Checked"
6. Go to Main Menu -> File -> Settings. Select Model Checker properties and uncheck all checkboxes except "Check for unresolved references" and press "Ok". Right-click project node in logical view and select "Check Project". The Model Checker tool will show you all the broken references you have. Correct them by hand.
7. Right-click project, select "Rebuild Project"
8. (Recommended, but not necessary) Restart MPS.

What the migration will do with my code?

Upgrade Persistence: Since MPS 1.5 we've changed the format in which models are stored on disk. Commonly this is a performance-related issue. This action will re-save all models and modules in your project in a new format.

Fix Module Dependencies: In MPS 1.5 we had some unclear dependencies especially in languages and language models. This action will reconsider module imports and change their structure (some devkits will be replaced with others)

Migrate Stub Models: If you tried to use different versions of the same library in older versions of MPS, you may know that MPS didn't handle such situation. References to stub models didn't contain any information about a place from where this model is. Now it also contains the ID of the module containing a model, so that you can use different versions of the same code with the only restriction - they should not be added as stubs to the same module. So, we need to update all the references to stub models from your code. MPS does it the following way: firstly, it tries to resolve the model in scope of current module using the old "partial" id. If it succeeds, the reference is then resolved. If this fails, MPS tries to resolve it globally and if this succeeds, add a corresponding module import and resolve the reference. Also, this action will correct model imports correspondingly.

Optimize Imports: This will remove dependencies on modules that don't exist in MPS anymore (and really unused dependencies, too)

Scripts: When we change language syntax, a common way of migrating the outdated code to the new syntax (where possible) is to provide a refactoring (we call this kind of refactorings "migration scripts" not to mix them with other types of refactorings) which will change old syntax to new one (if it's possible). This action will find all MigrationScript instances in MPS and execute those of them which have a "migration" type (there are also so-called "enhancement scripts", which will not be executed by this action).

Model Checker and manual references resolution: If you have used some internal API, or somewhy MPS couldn't resolve a stub reference or any other reference, you can correct them manually. This tool will help to find all such references.

Rebuild: After you have migrated your code (MPS models) to match 2.0 version, they should be rebuilt and recompiled to start working in MPS. That's what this step does.

Advanced cases

There was one change in MPS 2.0 for which we can't provide a common migration, so if you have used this feature, you are to correct your code by hand. But we suppose that nobody has actually used this feature till the moment.
So, the problem appears if you had created your own "attributes" (instances of `AnnotationLinkDeclaration`) with complex hierarchies (in case when your attributes are direct subconcepts of `BaseConcept`, the migration script will do everything for you, both in automatic and non-automatic scenario).
Migration for such hierarchical attributes is the following:

1. Ensure attribute concept described in `AnnotationLinkDeclaration` (`target`) is subconcept of one of `NodeAttribute`, `LinkAttribute` or `PropertyAttribute` depending on the attribute `stereotype`
2. Define `role` concept property in attribute concept as attribute `role` in `AnnotationLinkDeclaration`
3. If `sourceCardinality` was * define `multiple` concept property in attribute concept
4. Define `attributed` concept link from `source` in `AnnotationLinkDeclaration`

Migration to 2.0.1

1) To fix module dependencies, execute "Add missing imports" action on project